Theoretical Computer Science 401 (2008) 191–205

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Reversal complexity revisited

André Hernich *, Nicole Schweikardt

Institut für Informatik, Johann Wolfgang Goethe-Universität, Robert-Mayer-Straße 11-15, 60054 Frankfurt am Main, Germany

ARTICLE INFO

Article history:
Received 4 August 2006
Received in revised form 15 April 2008
Accepted 17 April 2008
Communicated by J. Diaz

Keywords:
Reversal-bounded and space-bounded Turing machines
Hierarchies of complexity classes

ABSTRACT

We study a generalized version of reversal bounded Turing machines where, apart from several tapes on which the number of head reversals is bounded by $r(n)$, there are several further tapes on which head reversals remain unrestricted, but *size* is bounded by $s(n)$ (where n denotes the input length). Recently [M. Grohe, C. Koch, N. Schweikardt, Tight lower bounds for query processing on streaming and external memory data, Theoretical Computer Science 380 (1–2) (2007) 199–217; M. Grohe, N. Schweikardt, Lower bounds for sorting with few random accesses to external memory, in: Proc. PODS'05, ACM Press, 2005, pp. 238–249], such machines were introduced as a formalization of a computation model that restricts random access to external memory and internal memory space. Here, each of the tapes with a restriction on the head reversals corresponds to an external memory device, and the tapes of restricted size model internal memory. We use $\mathrm{ST}(r(n), s(n), O(1))$ to denote the class of all problems that can be solved by deterministic Turing machines that comply to the above resource bounds. Similarly, $\mathrm{NST}(\cdots)$ and $\mathrm{RST}(\cdots)$, respectively, are used for the corresponding nondeterministic and randomized classes.

While previous papers focused on lower bounds for particular problems, including sorting, the set equality problem, and several query evaluation problems, the present paper addresses the relations between the $(\mathrm{R,N})\mathrm{ST}(\cdots)$-classes and classical complexity classes and investigates the structural complexity of the $(\mathrm{R,N})\mathrm{ST}(\cdots)$-classes. Our main results are (1) a trade-off between internal memory space and external memory head reversals, (2) correspondences between the $(\mathrm{R,N})\mathrm{ST}(\cdots)$ classes and "classical" time-bounded, space-bounded, reversal-bounded, and circuit complexity classes, and (3) hierarchies of $(\mathrm{R})\mathrm{ST}(\cdots)$-classes in terms of increasing numbers of head reversals on external memory tapes.

1. Introduction

Modern software and database technology uses clever heuristics to minimize the number of accesses to external memory and to prefer *streaming* over *random accesses* to external memory. There has also been a wealth of research on the design of so-called *external memory algorithms* (cf., e.g. [22,14]). The classes considered in *computational complexity theory*, however, usually do not take into account the existence of different storage media.

In [9,10] complexity classes for such a scenario were introduced. The two most significant resource bounds in this setting are imposed on the number of random accesses to external memory and the size of the internal memory. Our complexity classes are based on a standard multi-tape Turing machine. Some of the tapes of the machine, among them the input tape, represent external memory devices. They are unrestricted in size, but access to these tapes is restricted by allowing only a certain number $r(n)$ (where n denotes the input size) of reversals of the head directions. This may be seen as a way of (a)

* Corresponding author. Tel.: +49 69 798 28175; fax: +49 69 798 28334.
E-mail addresses: hernich@informatik.uni-frankfurt.de (A. Hernich), schweika@informatik.uni-frankfurt.de (N. Schweikardt).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.04.026

restricting the number of sequential scans and (b) restricting random access to these tapes, because each random access can be simulated by moving the head to the desired position on a tape, which involves at most two head reversals. The remaining tapes of the Turing machine represent the internal memory. Access to these internal memory tapes (i.e., the number of head reversals) is unlimited, but their size is bounded by a parameter $s(n)$. We let $\mathrm{ST}(r(n), s(n), t)$ denote the class of all problems that can be solved on such an $(r(n), s(n), t)$-bounded deterministic Turing machine, i.e., a Turing machine with t external memory tapes which, on inputs of size n, performs less than $r(n)$ head reversals on the external memory tapes, and uses at most space $s(n)$ on the internal memory tapes. Similarly, we use $\mathrm{NST}(r(n), s(n), t)$ and $\mathrm{RST}(r(n), s(n), t)$ for the corresponding nondeterministic, respectively, randomized classes. The acceptance criterion for the $\mathrm{RST}(\cdots)$ classes is defined in the same way as for the class RP of *randomized polynomial time*, i.e., "no"-instances of a decision problem are *always* rejected, whereas "yes"-instances are accepted with probability $\geq 1/2$.

With the "external memory" motivation in mind, we are mainly interested in classes where the number of head reversals and the internal memory size are comparably small, i.e., of size $o(n)$. Let us, however, emphasize that the main objective in introducing the $(\mathrm{N,R})\mathrm{ST}(\cdots)$-classes is *not* to provide a "realistic" computation model suitable for designing efficient external memory algorithms, but to provide robust complexity classes which reflect the existence of different storage media and which are *at least* as powerful as any "realistic" external memory computation model. Thus, *lower bounds* in terms of these complexity classes will immediately imply lower bounds for the "external memory complexity" of certain problems. For a more detailed discussion on the motivations for considering the $(\mathrm{N,R})\mathrm{ST}(\cdots)$-classes we refer the reader to the surveys [8,21].

Obviously, our $\mathrm{ST}(\cdots)$-classes are related to the *bounded reversal Turing machines*, which have been studied in classical complexity theory (see, e.g., [23,4]). However, in bounded reversal Turing machines, the number of head reversals is limited on *all* tapes, whereas in our model there is no such restriction on the internal memory tapes. Thus, a priori, the $\mathrm{ST}(\cdots)$-classes are considerably stronger than conventional bounded reversal classes.

In [10,9,7,3], *lower bounds* for particular problems, including the sorting problem, the set equality problem, and several query evaluation problems, have been shown for the deterministic and randomized $\mathrm{ST}(\cdots)$ classes. The relations between the $(\mathrm{R,N})\mathrm{ST}(\cdots)$-classes and classical complexity classes, as well as the structural complexity of the $(\mathrm{R,N})\mathrm{ST}(\cdots)$-classes remained as future tasks that are now addressed by the present paper, whose main results are

1. a trade-off between internal memory space and external memory head reversals, stating that internal memory can be compressed from size $s(n)$ to $O(1)$ at the expense of adding an extra factor $s(n)$ to the external memory head reversals (Theorem 3.1).
2. correspondences between the $(\mathrm{R,N})\mathrm{ST}(\cdots)$ classes and "classical" time-bounded, space-bounded, reversal-bounded, and circuit complexity classes. For example, we obtain that $\mathrm{NP} = \mathrm{NST}(O(1), O(\log n), O(1)) = \mathrm{NST}(O(\log n), O(1), O(1))$ (Corollary 4.4) and that $\textsc{Polylogspace} = \mathrm{ST}((\log n)^{O(1)}, (\log n)^{O(1)}, O(1))$ (Theorem 4.9).
3. hierarchies of $(\mathrm{R})\mathrm{ST}(\cdots)$-classes in terms of increasing numbers of head reversals on external memory tapes. E.g., for all $k \geq 2$, we obtain $\mathrm{RST}(O(\sqrt[k+1]{\log n}), O(\log n), O(1)) \subsetneq \mathrm{RST}(O(\sqrt[k]{\log n}), O(\log n), O(1))$ (Theorem 5.4). For the special case where only *one* external memory tape is available, we obtain for all functions $r(n) \in o(n/(\log n)^2)$ that adding one single extra head reversal leads to a strictly larger $\mathrm{ST}(\cdots)$ class (Theorem 5.5).

Organization. In Section 2 we formally introduce the $(\mathrm{R,N})\mathrm{ST}(\cdots)$ classes and summarize what has been known about these classes. Afterwards, in Section 3, we show a trade-off between internal memory size and external memory head reversals. Section 4 investigates the relations between the $(\mathrm{R,N})\mathrm{ST}(\cdots)$-classes and "classical" time-bounded, space-bounded, reversal-bounded, and circuit complexity classes. In Section 5 we prove hierarchies of deterministic and randomized $\mathrm{ST}(\cdots)$ classes in terms of increasing numbers of head reversals on external memory tapes. We close in Section 6 with a few concluding remarks.

2. Preliminaries

This section fixes some basic notation, gives a formal introduction of the $\mathrm{ST}(\cdots)$ complexity classes that were proposed in [10,9], and summarizes what is known about the inclusion structure of these classes.

We write $\mathbb{N}$ to denote the set of natural numbers excluding 0. All logarithms are to the base 2 unless otherwise stated.

As our basic model of computation, we use standard multi-tape nondeterministic Turing machines (NTMs, for short). The Turing machines we consider will have $t + u$ tapes. We call the first t tapes *external memory tapes* (and think of them as representing t external memory devices); the other u tapes are called *internal memory tapes*. The first external memory tape is always viewed as the (read/write) input tape.

Without loss of generality we assume that our Turing machines are normalized in such a way that in each step at most one of their heads moves to the left or to the right.

A finite run of an NTM T is a sequence $\rho = (\gamma_1, \ldots, \gamma_\ell)$ of configurations of T such that γ_1 is an initial configuration, γ_ℓ is a final configuration, and for all $i < \ell$, γ_{i+1} can be reached from γ_i in a single computation step.

Let T be an NTM and ρ a finite run of T. Let $i \geq 1$ be the number of a tape. We use $\mathrm{rev}(\rho, i)$ to denote the number of times the i-th head changes its direction in the run ρ. Furthermore, we let $\mathrm{space}(\rho, i)$ be the number of cells of tape i that are used by ρ.

Definition 2.1 (*(r, s, t)-bounded TM, [10,9]*). Let $r, s : \mathbb{N} \to \mathbb{N}$ and $t \in \mathbb{N}$. An NTM T is (r, s, t)-*bounded*, if every run ρ of T on an input of length n (for arbitrary $n \in \mathbb{N}$) satisfies the following conditions: (1) ρ is finite, (2) $1 + \sum_{i=1}^{t} \text{rev}(\rho, i) \leq r(n)$,[1] and (3) $\sum_{i=t+1}^{t+u} \text{space}(\rho, i) \leq s(n)$, where $t + u$ is the total number of tapes of T.

Definition 2.2 ($\text{ST}(\cdots)$ *and* $\text{NST}(\cdots)$ *classes, [10]*). Let $r, s : \mathbb{N} \to \mathbb{N}$ and $t \in \mathbb{N}$. A decision problem belongs to the class $\text{ST}(r, s, t)$ (resp., $\text{NST}(r, s, t)$), if it can be decided by a deterministic (resp., nondeterministic) (r, s, t)-bounded Turing machine.

Note that we put no restriction on the running time or the space used on the first t tapes of an (r, s, t)-bounded Turing machine. The following lemma shows that these parameters cannot get too large.

Lemma 2.3 (*[10,8]*). *Let $r, s : \mathbb{N} \to \mathbb{N}$ and $t \in \mathbb{N}$, and let T be an (r, s, t)-bounded NTM. Then for every run $\rho = (\gamma_1, \ldots, \gamma_\ell)$ of T on an input of size n we have $\ell \leq n \cdot 2^{O(r(n)\cdot(t+s(n)))}$ and $\sum_{i=1}^{t} \text{space}(\rho, i) \leq n \cdot 2^{O(r(n)\cdot(t+s(n)))}$.*

In [10,8], the lemma has only been stated and proved for deterministic Turing machines, but it is obvious that the same proof also applies to nondeterministic machines (to see this, note that, by definition, *every* run of an (r, s, t)-bounded Turing machine is finite).

In analogy to the definition of *randomized* complexity classes such as the class RP of randomized polynomial time (cf., e.g., [2,16]), we also consider the randomized version $\text{RST}(\cdots)$ of the $\text{ST}(\cdots)$ and $\text{NST}(\cdots)$ classes. The following definition of randomized Turing machines formalizes the intuition that in each step a coin can be tossed to determine which particular successor configuration is chosen in this step. For a configuration γ of an NTM T, we write $\text{Next}_T(\gamma)$ to denote the set of all configurations γ' that can be reached from γ in a single computation step. Each such configuration $\gamma' \in \text{Next}_T(\gamma)$ is chosen with uniform probability, i.e., $\Pr(\gamma \to_T \gamma') = 1/|\text{Next}_T(\gamma)|$. For a run $\rho = (\gamma_1, \ldots, \gamma_\ell)$, the probability $\Pr(\rho)$ that T performs run ρ is the product of the probabilities $\Pr(\gamma_i \to_T \gamma_{i+1})$, for all $i < \ell$. For an input word w, the probability $\Pr(T \text{ accepts } w)$ that T accepts w is the sum of $\Pr(\rho)$ for all accepting runs ρ on input w.

We say that a decision problem L is solved by a $(\frac{1}{2}, 0)$-RTM if, and only if, there is an NTM T such that every run of T has finite length, and the following is true for all input instances w: If $w \in L$, then $\Pr(T \text{ accepts } w) \geq \frac{1}{2}$; if $w \notin L$, then $\Pr(T \text{ accepts } w) = 0$.

Definition 2.4 ($\text{RST}(\cdots)$ *Classes, [7]*). Let $r, s : \mathbb{N} \to \mathbb{N}$ and $t \in \mathbb{N}$. A decision problem L belongs to the class $\text{RST}(r, s, t)$ if it can be solved by a $(\frac{1}{2}, 0)$-RTM that is (r, s, t)-bounded.

As a straightforward observation one obtains:

Proposition 2.5. *For all $r, s : \mathbb{N} \to \mathbb{N}$ and $t \in \mathbb{N}$, $\text{ST}(r, s, t) \subseteq \text{RST}(r, s, t) \subseteq \text{NST}(r, s, t)$.*

For classes R and S of functions we let

$$\text{ST}(R, S, t) := \bigcup_{r\in R, s\in S} \text{ST}(r, s, t) \quad \text{and} \quad \text{ST}(R, S, O(1)) := \bigcup_{t\in\mathbb{N}} \text{ST}(R, S, t).$$

The classes $\text{NST}(R, S, t)$, $\text{RST}(R, S, t)$, $\text{NST}(R, S, O(1))$, and $\text{RST}(R, S, O(1))$ are defined in the analogous way.

As usual, for every complexity class C, we write co-C to denote the class of all decision problems whose *complements* belong to C. Note that the $\text{RST}(\cdots)$-classes consist of decision problems that can be solved by randomized algorithms that *always* reject "no"-instances and that accept "yes"-instances with probability $\geq \frac{1}{2}$. In contrast to this, the co-$\text{RST}(\cdots)$-classes consist of problems that can be solved by randomized algorithms that *always* accept "yes"-instances and that reject "no"-instances with probability $\geq \frac{1}{2}$.

From Lemma 2.3, one immediately obtains

Corollary 2.6. *Let $r, s : \mathbb{N} \to \mathbb{N}$ and $t \in \mathbb{N}$. Then*

$$(\text{N,R})\text{ST}(r, s, t) \subseteq (\text{N,R})\text{TIME}(2^{O(r(n)\cdot(t+s(n)))+\log n}),$$

where $\text{RTIME}(2^{O(r(n)\cdot(t+s(n)))+\log n})$ denotes the class of all decision problems that can be solved by a $(\frac{1}{2}, 0)$-RTM that has time bound $2^{O(r(n)\cdot(t+s(n)))+\log n}$.

In particular, whenever $r(n) \cdot s(n) \in O(\log n)$, we have

$$\text{ST}(r, s, O(1)) \subseteq \text{PTIME}, \qquad \text{RST}(r, s, O(1)) \subseteq \text{RP}, \qquad \text{NST}(r, s, O(1)) \subseteq \text{NP},$$

where RP denotes randomized polynomial time, i.e., the class of all decision problems that can be solved by a polynomial time bounded $(\frac{1}{2}, 0)$-RTM.

[1] It is convenient for technical reasons to add 1 to the number $\sum_{i=1}^{t} \text{rev}(\rho, i)$ of changes of the head direction. As defined here, $r(n)$ thus bounds the number of sequential scans of the external memory tapes rather than the number of changes of head directions.

Separation results are known for, e.g., the following decision problems:

MULTISET-EQUALITY
Instance: $v_1\#\cdots v_m\#v'_1\#\cdots v'_m\#$,
where $m \geq 1$ and $v_1, \ldots, v_m, v'_1, \ldots, v'_m \in \{0,1\}^*$.
Problem: Decide if the multisets $\{v_1, \ldots, v_m\}$ and $\{v'_1, \ldots, v'_m\}$ are equal (i.e., they contain the same elements with the same multiplicities).

SHORT-SET-EQUALITY
Instance: $v_1\#\cdots v_m\#v'_1\#\cdots v'_m\#$,
where $m \geq 1$ and $v_1, \ldots, v_m, v'_1, \ldots, v'_m \in \{0,1\}^{2\cdot\log m}$.
Problem: Decide if $\{v_1, \ldots, v_m\} = \{v'_1, \ldots, v'_m\}$ as sets (i.e., disregarding multiplicities of elements).

One of the main results of [7] shows:

Theorem 2.7 *([7]).* MULTISET-EQUALITY *and* SHORT-SET-EQUALITY

(a) *do not belong to* $\mathrm{RST}(r, s, O(1))$,
whenever r and s are functions with $r(n) \in o(\log n)$ *and* $s(n) \in o(\sqrt[4]{n}/r(n))$.
(b) *belong to* $\mathrm{NST}(3, O(\log n), 2)$ *and to* $\mathrm{ST}(O(\log n), O(1), O(1))$.

Furthermore, the MULTISET-EQUALITY *problem belongs to* $\text{co-RST}(2, O(\log n), 1)$.

This immediately leads to the following separations between the deterministic, randomized, and nondeterministic $\mathrm{ST}(\cdots)$ classes:

Corollary 2.8 *([7]).*
Let r and s be functions with $r(n) \in o(\log n)$ *and* $s(n) \in o(\sqrt[4]{n}/r(n)) \cap \Omega(\log n)$. *Then,*

(a) $\mathrm{RST}(O(r), O(s), O(1)) \neq \text{co-RST}(O(r), O(s), O(1))$,
(b) $\mathrm{ST}(O(r), O(s), O(1)) \subsetneq \mathrm{RST}(O(r), O(s), O(1)) \subsetneq \mathrm{NST}(O(r), O(s), O(1))$.

3. Trade-off between internal space and external head reversals

In this section we show that internal memory tapes can be simulated by head reversals on external memory tapes:

Theorem 3.1. *For all* $t \in \mathbb{N}$ *and all functions* $r, s\colon \mathbb{N} \to \mathbb{N}$ *with* $r(n)\cdot s(n) \in \Omega(\log n)$,

$$\begin{aligned}\mathrm{ST}(r, s, t) &\subseteq \mathrm{ST}(O(r\cdot s), O(1), t+2),\\ \mathrm{RST}(r, s, t) &\subseteq \mathrm{RST}(O(r\cdot s), O(1), t+2),\\ \mathrm{NST}(r, s, t) &\subseteq \mathrm{NST}(O(r\cdot s), O(1), t+2).\end{aligned}$$

Proof. We first prove the deterministic case. To this end, let M be a deterministic (r, s, t)-bounded Turing machine, where $r, s\colon \mathbb{N} \to \mathbb{N}$ are functions with $r(n)\cdot s(n) \in \Omega(\log n)$. Without loss of generality, M has just one internal memory tape, and M's tape alphabet Σ has at least two symbols, one of them being the blank symbol $\square \in \Sigma$. We show how to simulate M by a deterministic $(O(r\cdot s), O(1), t+2)$-bounded Turing machine M'.

Let x be an input of length n. Our simulation proceeds in *meta-steps*, each of which simulates one step of M on input x. That is, starting from the initial configuration of M on input x, each meta-step computes the successor configuration of the current configuration of M. Throughout the simulation, the current configuration of M is represented by

- the first t external memory tapes of M' (i.e., after the ith meta-step, the contents and head positions of these tapes will be the same as after the ith step of the computation of M on input x), and
- a string of the form w_1qw_2, where q is the current state, w_1w_2 is a string of length at most $s(n)$ representing the contents of the internal memory tape, $|w_2| \geq 1$, and $|w_1| + 1$ is the position of the internal memory tape's head (this means that the head is placed on the first symbol of w_2).

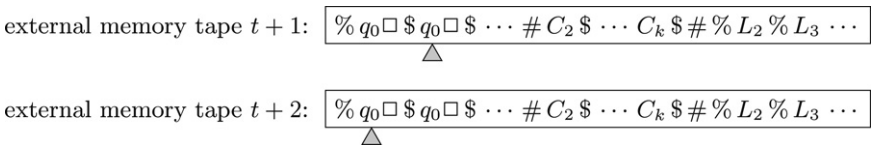
We call w_1qw_2 an *internal configuration* of M. The internal configuration $q_0\square$, where q_0 is the start state of M and $\square$ is the blank symbol, is the *initial internal configuration* of M.

Assume for the moment that the last two external memory tapes of M' contain sufficiently many copies of the following string

$$L = \underbrace{C_1\,\$\,C_1\,\$\,\cdots\,C_1\,\$}_{2k\text{ times}}\;\#\;\underbrace{C_2\,\$\,C_2\,\$\,\cdots\,C_2\,\$}_{2k\text{ times}}\;\#\;\cdots\;\underbrace{C_k\,\$\,C_k\,\$\,\cdots\,C_k\,\$}_{2k\text{ times}}\;\#,$$

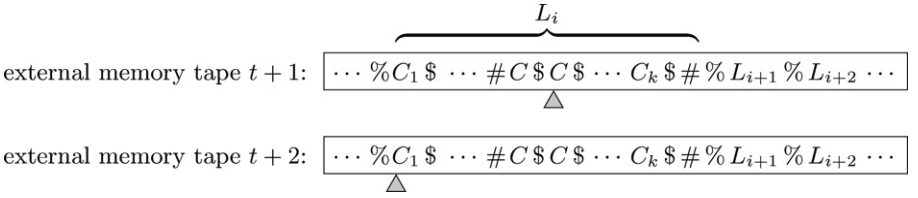
where $C_1, C_2, \ldots, C_k$ is an enumeration of all possible *internal* configurations of M on an input of length n, C_1 is the initial internal configuration of M, and the symbols \$ and # do not occur in Σ (we will explain later how to produce these copies of L). [2] More precisely, let both external memory tapes contain the string $\%L_1\%L_2\%\cdots\%L_\ell\%$ for some sufficiently large ℓ, where $L_1 = L_2 = \cdots = L_\ell = L$, and the symbol % does not occur in Σ. Then the simulation works as follows.

Initialization step: In the initialization step, we move the head of tape $t+1$ to the first symbol of the second copy of $C_1 = q_0\Box$ in L_1, and the head of tape $t+2$ to the first symbol of L_1:



Note that C_1 together with the contents and head positions of the first t external memory tapes of M' represents the initial configuration of M on input x.

Meta-step $i \geq 1$: For the ith meta-step, assume that the head of tape $t+1$ is placed on the first symbol of the second copy of an internal configuration C in L_i, and that the head of tape $t+2$ is placed on the first symbol of L_i:



Let γ be the configuration of M that is represented by C and the first t external memory tapes of M', and let γ' be the successor configuration of γ. Moreover, let C' be the *internal* configuration of M that corresponds to the state and the configuration of the internal memory tape in γ'. Our aim is to move the head of external memory tape $t+1$ to the first symbol of the second copy of C' in L_{i+1}, to move the head of external memory tape $t+2$ to the first symbol of L_{i+1}, and to modify the first t external memory tapes such that C' together with these tapes represents γ'. This can be achieved as follows.

1. *Update the first t external memory tapes:* The head movements and the symbols to be written can be determined by reading the second copy of C in L_i on tape $t+1$ and the symbols at the head positions of the first t external memory tapes.
2. *Find C' in L_i on tape $t+2$:* We use the next k copies of C in L_i on tape $t+1$ to compare C symbol-wise with the first copy of each C_j in L_i on tape $t+2$ until we find C'. We always read three symbols of C in advance, so that, when we come to the state and the symbol following the state, we can check without any head reversal whether C_j is modified according to the transition function of M. Finally, we move the head of tape $t+2$ to the leftmost symbol of the next copy of C' in L_i.
3. *Find C' in L_{i+1} on tape $t+1$:* We use the next k copies of C' in L_i on tape $t+2$ to compare C' symbol-wise with the first copy of each C_j in L_{i+1} on tape $t+1$ until we find C'. Finally, we move the head of tape $t+1$ to the first symbol of the second copy of C' in L_{i+1}, and the head of tape $t+2$ to the beginning of L_{i+1}.

This finishes the description of the ith meta-step. Clearly, if the list $\%L_1\%L_2\%\cdots\%L_\ell\%$ is given on external memory tapes $t+1$ and $t+2$, and ℓ is at least as large as the number of steps in the computation of M on input x, then we can compute the final configuration of M on input x, and decide whether M accepts x or not, with a single left-to-right scan on external memory tapes $t+1$ and $t+2$, and no internal memory tapes, while performing exactly the same head movements as M on the first t external memory tapes.

To finish the proof for the deterministic case, we explain how the list $\%L_1\%L_2\%\cdots L_\ell\%$ can be computed. To this end, the following definition is very convenient: an *internal configuration with space m* is an internal configuration of the form $w_1 q w_2$ with $|w_1 w_2| = m$. The list is now computed "on the fly", starting with a list of all internal configurations with space 1, and adding more copies to the list, or adding internal configurations with more space, if necessary. More precisely, M' starts with a list $\%L^{(1)}\%$ that contains just a single copy of the string

$$L^{(1)} = \underbrace{C_1^{(1)}\$C_1^{(1)}\$\cdots C_1^{(1)}\$}_{2k_1\text{ times}}\;\#\;\underbrace{C_2^{(1)}\$C_2^{(1)}\$\cdots C_2^{(1)}\$}_{2k_1\text{ times}}\;\#\;\cdots\;\underbrace{C_{k_1}^{(1)}\$C_{k_1}^{(1)}\$\cdots C_{k_1}^{(1)}\$}_{2k_1\text{ times}}\;\#,$$

[2] The $2k$ copies of each internal configuration C_j in L are used here merely to simplify our description of producing the copies of L; for the following simulation, blocks of $k+2$ copies of each internal configuration would suffice.

with all possible internal configurations $C_1^{(1)}, C_2^{(1)}, \ldots, C_{k_1}^{(1)}$ with space 1. Such a string can be produced without any head reversal and only constant internal memory.

Assume now for $i \geq 1$ that at the beginning of the ith meta-step, external memory tapes $t+1$ and $t+2$ contain the list $\%L_1^{(i)}\%L_2^{(i)} \ldots \%L_{\ell_i}^{(i)}\%$ with $L_1^{(i)} = \cdots = L_{\ell_i}^{(i)} = L^{(i)}$, where

$$L^{(i)} = \underbrace{C_1^{(i)}\$C_1^{(i)}\$\cdots C_1^{(i)}\$}_{2k_i \text{ times}} \# \underbrace{C_2^{(i)}\$C_2^{(i)}\$\cdots C_2^{(i)}\$}_{2k_i \text{ times}} \# \cdots \underbrace{C_{k_i}^{(i)}\$C_{k_i}^{(i)}\$\cdots C_{k_i}^{(i)}\$}_{2k_i \text{ times}} \#$$

contains all possible internal configurations $C_1^{(i)}, C_2^{(i)}, \ldots, C_{k_i}^{(i)}$ with space m, for some $m \geq 1$. There are three possible cases:

1. The ith meta-step can be performed as described above.
2. The number ℓ_i of currently available copies of $L^{(i)}$ is too small.
3. The internal configurations are too short, i.e., the current internal configuration C uses m tape cells, but its successor C' uses $m+1$ tape cells (and thus does not occur in the currently available list $L^{(i)}$).

In case 2 we perform a *doubling step*, and in case 3 an *extension step*, as described below.

Doubling step: If in the ith meta-step, M' passes the last % on external memory tape $t+1$ or $t+2$, it doubles the number of copies of $L^{(i)}$ on external memory tapes $t+1$ and $t+2$ before continuing the simulation. This is done as follows: The current positions on both tapes are marked so that M' can return to these positions later. To double the number of copies of $L^{(i)}$, M' first appends the list on external memory tape $t+1$ to the list on external memory tape $t+2$; the result is a list that contains $2 \cdot \ell_i$ copies of $L^{(i)}$. Then M' replaces the list on external memory tape $t+1$ with the new list on external memory tape $t+2$. The doubling step needs a constant number of reversals and constant internal memory.

Extension step: If the successor C' of the current internal configuration C in the ith meta-step is an internal configuration with space $m+1$, then M' replaces each copy of $L^{(i)}$ on external memory tapes $t+1$ and $t+2$ by a string $L^{(i+1)}$ that contains all possible internal configurations with space $m+1$. This is done as follows. The current internal configurations on both tapes, and the positions in these configurations, are marked so that M' can return to these positions in the corresponding extended internal configurations later. Each internal configuration with space $m+1$ is generated from one of the internal configurations $C_j^{(i)}$ in $L^{(i)}$. More precisely, for each internal configuration $C_j^{(i)} = w_1 q x w_2$ in $L^{(i)}$, and each tape symbol $y \in \Sigma$, M' generates internal configurations $w_1 q x w_2 y$ and $w_1 x q w_2 y$, so that after the extension step, $L^{(i+1)}$ contains exactly

$$k_{i+1} := 2 \cdot |\Sigma| \cdot k_i \tag{1}$$

internal configurations with space $m+1$ (some configurations may occur in the list for several times, but this does not do any harm here).

The extension step can be implemented with a constant number of reversals, and constant internal memory, as follows:

1. With one pass over external memory tapes $t+1$ and $t+2$, we replace the list on external memory tape $t+2$ by a list $\%L'\%L' \cdots \%L'\%$ of ℓ_i copies of

$$L' = \underbrace{C_1'\,\$\,C_1'\,\$\,\cdots\,C_1'\,\$}_{2k_i \text{ times}} \# \underbrace{C_2'\,\$\,C_2'\,\$\,\cdots\,C_2'\,\$}_{2k_i \text{ times}} \# \cdots \underbrace{C_{k_i}'\,\$\,C_{k_i}'\,\$\,\cdots\,C_{k_i}'\,\$}_{2k_i \text{ times}} \#,$$

   where $C_j' = C_j^{(i)}\square$ for each $j \in \{1, 2, \ldots, k_i\}$. Each C_j' will be used as a template for an internal configuration with space $m+1$ in the following step.
2. With one pass over external memory tapes $t+1$ and $t+2$, we replace the list on external memory tape $t+1$ by

$$\underbrace{\%\ \underbrace{\star\ \star\ \cdots\ \star}_{2\cdot|\Sigma|\cdot|L'| \text{ times}}\ \%\ \underbrace{\star\ \star\ \cdots\ \star}_{2\cdot|\Sigma|\cdot|L'| \text{ times}}\ \%\ \cdots\ \%\ \underbrace{\star\ \star\ \cdots\ \star}_{2\cdot|\Sigma|\cdot|L'| \text{ times}}\ \%}_{\ell_i \text{ segments of stars}},$$

   where $\star$ does not occur in Σ. This can be achieved by writing $2|\Sigma|$ stars for each symbol of L'. Let us call a segment of stars a *segment*.
   For each symbol $y \in \Sigma$, we make two passes over external memory tapes $t+1$ and $t+2$, and fill each segment on external memory tape $t+1$ as follows. In the first pass, we add to each segment a modified version of L', in which each internal configuration $C_j' = C_j^{(i)}\square$ is replaced by $C_j^{(i)}y$. In the second pass, we add to each segment a modified version of L', in which each internal configuration $C_j' = w_1 q x w_2 \square$ is replaced by $w_1 x q w_2 y$. Thus, after $2|\Sigma|$ passes over external memory tapes $t+1$ and $t+2$, external memory tape $t+1$ contains a list $\%L''\%L'' \cdots \%L''\%$ of ℓ_i copies of

$$L'' = \underbrace{C_1''\,\$\,C_1''\,\$\,\cdots\,C_1''\,\$}_{2k_i \text{ times}} \# \underbrace{C_2''\,\$\,C_2''\,\$\,\cdots\,C_2''\,\$}_{2k_i \text{ times}} \# \cdots \underbrace{C_{k_{i+1}}''\,\$\,C_{k_{i+1}}''\,\$\,\cdots\,C_{k_{i+1}}''\,\$}_{2k_i \text{ times}} \#,$$

   where $C_1'', \ldots, C_{k_{i+1}}''$ is an enumeration of all internal configurations of M with space $m+1$ (possibly with repetitions).

3. Recall from (1) that $k_{i+1} = 2|\Sigma|k_i$. With $2|\Sigma|+1$ passes over external memory tapes $t+1$ and $t+2$, we replace the list on external memory tape $t+2$ by a list $\%L^{(i+1)}\%L^{(i+1)}\%\cdots\%L^{(i+1)}\%$ of ℓ_i copies of

$$L^{(i+1)} = \underbrace{C''_1\,\$\,C''_1\,\$\,\cdots\,C''_1\,\$}_{2k_{i+1}\text{ times}}\,\#\,\underbrace{C''_2\,\$\,C''_2\,\$\,\cdots\,C''_2\,\$}_{2k_{i+1}\text{ times}}\,\#\,\cdots\,\underbrace{C''_{k_{i+1}}\,\$\,C''_{k_{i+1}}\,\$\,\cdots\,C''_{k_{i+1}}\,\$}_{2k_{i+1}\text{ times}}\,\#.$$

This is done in a similar way as in the previous step: in a first pass we produce a "template" by writing $2|\Sigma|$ stars for each symbol of a block $C''_j\ \$\ C''_j\ \$\ \cdots\ C''_j\ \$$ of $2k_{i+1}$ copies of an internal configuration C''_j. In $2|\Sigma|$ further passes we fill each segment of stars with $2|\Sigma|$ copies of the corresponding block. Finally, we copy the list from external memory tape $t+2$ to external memory tape $t+1$.

This finishes the description of the extension step. The extension step needs a constant number of reversals and constant internal memory.

By Lemma 2.3, M performs on an input of length n at most $\ell \leq 2^{O(r(n)\cdot s(n))}$ computation steps (recall that $r(n)\cdot s(n) = \Omega(\log n)$ by the assumption of the theorem). Therefore, we need at most $O(\log \ell) = O(r(n)\cdot s(n))$ doubling steps during the simulation. Moreover, since M uses space at most $s(n)$ on its internal memory tape, we need at most $s(n)$ extension steps during the simulation. Since every single extension step and doubling step requires only a constant number of head reversals, the entire simulation uses $O(r(n)\cdot s(n))$ head reversals on external memory tapes $t+1$ and $t+2$. Together with the $r(n)$ reversals on the first t external memory tapes, this yields $O(r(n)\cdot s(n))$ head reversals on the external memory tapes. This shows that M can be simulated by a deterministic $(O(r\cdot s), O(1), t+2)$-bounded Turing machine, and the proof of Theorem 3.1 for the deterministic case is complete.

The proofs for the randomized and nondeterministic cases are the same, except that in each meta-step, the successor configuration γ' of the current configuration γ of M is determined in a randomized way, respectively, in a nondeterministic way. In the randomized case, we choose a successor configuration by randomly selecting a possible transition (based on the current state and the symbols at the current head positions of the first t external memory tapes) and looking for the corresponding successor configuration. So the probability that we choose a successor configuration γ' of γ during the simulation is precisely the probability that γ yields γ' in a run of M. Therefore, the probability that x is accepted in the simulation is the same as the probability that x is accepted by M. □

Remark 3.2. The proof of Theorem 3.1 is inspired by the proof of Chen and Yap's Theorem 8 in [4]. Their theorem states that computations in DSpace(s) can be simulated by deterministic 2-tape Turing machines with $O(s)$ head reversals, provided that $s(n) \in \Omega(\log n)$ and s is *reversal-constructible* (i.e., given n in unary, a string of length $s(n)$ can be produced by a multi-tape Turing machine that performs $O(s(n))$ head reversals). Note, however, that our proof does not need an assumption on any kind of reversal-constructibility of r or s, and that we have to deal with computations that, in addition to the space-bounded internal memory tapes, also involves the external memory tapes (which are bounded in the number of head-reversals, but unbounded in space).

Remark 3.3. Theorem 3.1 states that

$$(\text{N,R})\text{ST}(r, s, t) \subseteq (\text{N,R})\text{ST}(O(r\cdot s), O(1), t+2),$$

provided $r(n)\cdot s(n) \in \Omega(\log n)$. Inclusion in the other direction fails in general, at least for the deterministic and the randomized case. For example, by Theorem 3.1 we have

$$(\text{R})\text{ST}(O(1), O(\log n), O(1)) \subseteq (\text{R})\text{ST}(O(\log n), O(1), O(1)).$$

However, $(\text{R})\text{ST}(O(\log n), O(1), O(1)) \not\subseteq (\text{R})\text{ST}(O(1), O(\log n), O(1))$ since, according to Theorem 2.7, the Multiset-Equality problem belongs to $\text{ST}(O(\log n), O(1), O(1))$, but *not* to $\text{RST}(O(1), O(\log n), O(1))$.

4. "External memory" vs. classical complexity classes

In this section we clarify the relations between the ST(···) classes and "classical" complexity classes such as time- or space-bounded classes, reversal-bounded classes, and circuit complexity classes. Fig. 1 at the end of Section 4 visualizes some of the present section's results.

4.1. Time-bounded complexity classes

Recall that a function $f\colon \mathbb{N} \to \mathbb{N}$ is called *space-constructible* (cf., e.g., [16]) if there is a deterministic Turing machine that, given $n \in \mathbb{N}$ in unary on its (read-only) input tape, writes $f(n)$ in unary on its output tape and uses $O(f(n))$ space on its work tapes.

Theorem 4.1. *For all space-constructible functions $t\colon \mathbb{N} \to \mathbb{N}$, we have*

$$\text{NTime}(2^{t(n)}) \subseteq \text{NST}(3, O(t(n)), 2).$$

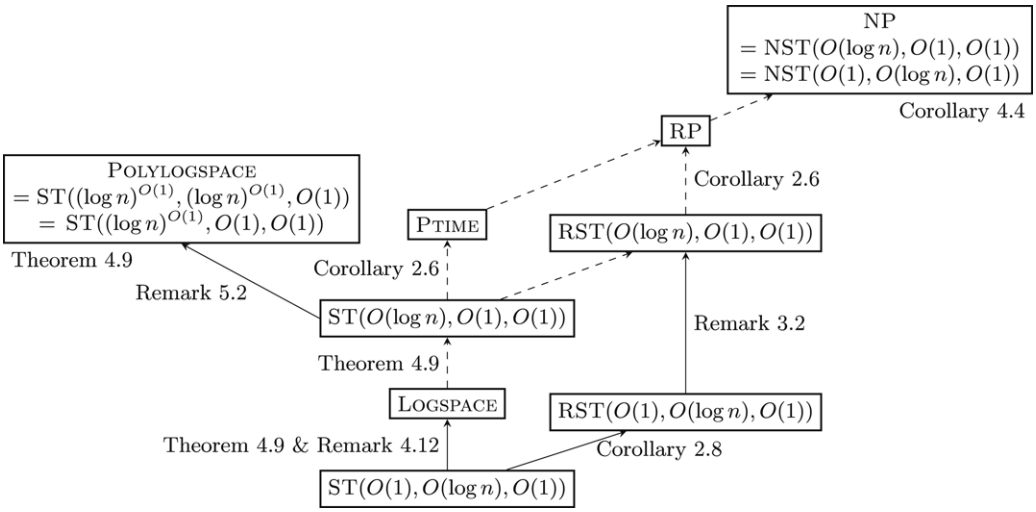


Fig. 1. Visualization of some of the relations between (R,N)ST(···) and classical complexity classes. Solid arrows indicate *strict* inclusions.

Proof. Let M be a nondeterministic Turing machine that performs at most $2^{t(n)}$ computation steps on inputs of length n. We describe how to simulate M by a $(3, O(t(n)), 2)$-bounded nondeterministic Turing machine. Let x be an input of length n. The machine nondeterministically guesses a number $k \leq 2^{t(n)}$ and writes (the same) sequence of configurations $C_0, C_1, \ldots, C_k$ on both external memory tapes. It does so without any head reversal and by using that $t(n)$ is space-constructible. If C_k is rejecting, then the machine rejects, too. If C_k is accepting, then the machine checks in a backward scan of both external memory tapes that for every $i \in \{0, 1, \ldots, k-1\}$, C_{i+1} is a successor configuration of C_i, and that C_0 is the start configuration of M on input x. The first condition can be checked without any head reversal while scanning C_i and C_{i+1} on separate tapes, and the latter one can be checked while scanning C_0 and x on separate tapes. The machine accepts if, and only if, both conditions are satisfied. □

Together with Theorem 3.1 and Corollary 2.6, we obtain the following:

Corollary 4.2. *For all space-constructible functions $t(n) \in \Omega(\log n)$, we have*

$$\begin{aligned}\text{NTime}(2^{O(t(n))}) &= \text{NST}(3, O(t(n)), 2)\\ &= \text{NST}(O(1), O(t(n)), O(1)) = \text{NST}(O(t(n)), O(1), O(1)).\end{aligned}$$

Proof.

$$\begin{aligned}\text{NTime}(2^{O(t(n))}) &\subseteq \text{NST}(O(1), O(t(n)), O(1)) && \text{(Theorem 4.1)}\\ &\subseteq \text{NST}(O(t(n)), O(1), O(1)) && \text{(Theorem 3.1)}\\ &\subseteq \text{NTime}(2^{O(t(n))}) && \text{(Corollary 2.6)}\quad \square\end{aligned}$$

Remark 4.3. Corollary 4.2 tells us that internal memory space can freely be traded for external memory head reversals, and vice versa, on *nondeterministic* machines. Contrast this with Remark 3.3 which states that for *deterministic* and *randomized* machines, external memory head reversals are strictly more powerful than internal memory space. I.e.:

$$\begin{aligned}&\text{NST}(O(1), O(\log n), O(1)) = \text{NST}(O(\log n), O(1), O(1)), && \text{but}\\ &\text{RST}(O(1), O(\log n), O(1)) \subsetneq \text{RST}(O(\log n), O(1), O(1)) && \text{and}\\ &\text{ST}(O(1), O(\log n), O(1)) \subsetneq \text{ST}(O(\log n), O(1), O(1)).\end{aligned}$$

Note that Corollary 4.2 gives us, in particular, a characterization of the class NP:

Corollary 4.4. $\text{NP} = \text{NST}(O(1), O(\log n), O(1)) = \text{NST}(O(\log n), O(1), O(1))$.

Remark 4.5. The classes Ptime and RP cannot be characterized in the analogous way: Obviously, the Multiset-Equality problem belongs to Ptime $\subseteq$ RP but, due to Theorem 2.7, it does *not* belong to $\text{RST}(O(1), O(\log n), O(1)) \supseteq \text{ST}(O(1), O(\log n), O(1))$. Thus, together with Corollary 2.6 we obtain

$$\text{RP} \supsetneq \text{RST}(O(1), O(\log n), O(1)) \quad \text{and} \quad \text{Ptime} \supsetneq \text{ST}(O(1), O(\log n), O(1)).$$

It remains a future task to determine whether the inclusions

$$\text{RP} \supseteq \text{RST}(O(\log n), O(1), O(1)) \quad \text{and} \quad \text{Ptime} \supseteq \text{ST}(O(\log n), O(1), O(1))$$

are strict. However, even an inclusion of the form

$$\text{Ptime} \subseteq \text{ST}((\log n)^{O(1)}, O(1), O(1)) \qquad (\star)$$

seems rather unlikely, since ($\star$), together with the next subsection's Theorem 4.9, would imply that PTIME $\subseteq$ POLYLOGSPACE (where POLYLOGSPACE denotes the class of all decision problems that can be solved in space $(\log n)^{O(1)}$).

Currently, the best simulation of time-bounded computations by deterministic reversal-bounded computations seems to be Liśkiewicz's result [13] that, for any function T with $T(n) \geq n$, $\mathrm{DTIME}(T) \subseteq \mathrm{ST}(O(\sqrt{T}), O(1), O(1))$.

4.2. Reversal-bounded and space-bounded complexity classes

Different versions of *reversal-bounded* computation models have been studied in the literature, cf., e.g., [23,4,18,17,19]. The model that is closest to our $\mathrm{ST}(\cdots)$ classes is Chen and Yap's [4] class $\mathrm{DREVERSAL}(r)$ which consists of all decision problems that can be solved by a deterministic multi-tape Turing machine that, on an input x of length n, performs at most $O(r(n))$ head reversals on all its tapes, provided that x is a "yes"-instance of the decision problem — on "no"-instances, Chen and Yap's model does not impose any resource bounds. However, as pointed out in [4], it does not make any difference to also impose the resource bounds for "no"-instances, provided that the function r is "reasonable" in the sense that it is *reversal-constructible*, i.e., there is a deterministic 2-tape Turing machine which, given $n \in \mathbb{N}$ in unary, generates $r(n)$ in unary on one of its tapes and uses $O(r(n))$ head reversals in total. It is noted in [4] that many complexity functions, including $\log n$, $(\log n)^i$, and n^i, are in fact reversal-constructible.

So, essentially, the class $\mathrm{DREVERSAL}(r)$ corresponds to our class $\mathrm{ST}(O(r), O(1), O(1))$, where only *constant* internal memory is available. Translated into the present paper's terminology, one of the main results of [4] is:

Theorem 4.6 (*Chen and Yap [4]*). *Let s be a reversal-constructible function and let r be an arbitrary function with $r(n), s(n) \in \Omega(\log n)$. Then,*

$$\mathrm{DSPACE}(s) \subseteq \mathrm{ST}(O(s), O(1), 2) \quad \textit{and} \quad \mathrm{ST}(O(r), O(1), O(1)) \subseteq \mathrm{DSPACE}(r^2).$$

From this, together with Theorem 3.1, one immediately obtains the following:

Corollary 4.7. *For all functions r and s with $r(n)\cdot s(n) \in \Omega(\log n)$, we have*

$$\mathrm{ST}(r, s, O(1)) \subseteq \mathrm{DSPACE}(r^2 \cdot s^2).$$

By giving a *direct* proof, we can slightly improve this to

Lemma 4.8. *For all functions r and s with $r(n)\cdot s(n) \in \Omega(\log n)$, we have*

$$\mathrm{ST}(r, s, O(1)) \subseteq \mathrm{DSPACE}(r^2 \cdot s).$$

Proof. Let r and s be functions with $r(n)\cdot s(n) \in \Omega(\log n)$, let M be a $(r(n), s(n), t)$-bounded deterministic Turing machine, and let u be the number of internal memory tapes of M. Let x be an input of length n. We describe how to determine the final state of M on input x in space $O(r(n)^2 \cdot s(n))$.

The idea is to construct a space-bounded machine that simulates M by re-computing the content of a tape cell of M every time it encounters this tape cell. Here, we proceed by induction on the number of head reversals. Assume that we have already constructed a procedure that can compute the state and content of M's tapes on input x after the kth head reversal in space $O(f_k(n))$, for an appropriate function f_k. Then we can compute the state and content of M's tapes after the $(k+1)$st head reversal in space $O(f_k(n) + r(n)\cdot s(n))$ by simulating M, starting in the state after the kth head reversal, where this state and all tape contents are determined as needed by computing the corresponding bits by the procedure for k. Here, the extra space of size $O(r(n)\cdot s(n))$ is needed to store the current head positions on all tapes (recall from Lemma 2.3 that M can visit at most $2^{O(r(n)\cdot s(n))+\log n}$ cells on each tape).

More formally, define a function S such that for every input string x and every $k \in \mathbb{N}$, $S(x, k)$ is the state of M on input x directly after the kth head reversal (if M on input x performs less than k head reversals, then $S(x, k) = S(x, k-1)$); $S(x, 0)$ is the start state of M. We use two other functions, P_i and T_i, for each tape $i \in \{1, \ldots, t+u\}$ of M. For every input string x, every $k \in \mathbb{N}$, and every $j \in \mathbb{N}$, $P_i(x, k)$ is the position of the head of the ith tape of M on input x after the kth head reversal, and $T_i(x, k, j)$ is the jth symbol on tape i of M on input x after the kth head reversal (again, if M on input x performs less than k head reversals, then $P_i(x, k) = P_i(x, k-1)$ and $T_i(x, k, j) = T_i(x, k-1, j)$); for $k = 0$, P_i and T_i describe the situation at the beginning of the computation.

To compute $S(x, k)$, we start simulating M in state $S(x, k-1)$ such that the head of tape i is positioned on cell $P_i(x, k-1)$. Whenever M reads cell j on tape i, we compute $T_i(x, k-1, j)$ and continue the simulation until M encounters a final state or a head reversal occurs. Then $S(x, k)$ is the last state of M in the simulation. P_i and T_i can be computed in a similar way: $P_i(x, k)$ is simply the position of the ith head at the end of this simulation, whereas $T_i(x, k, j)$ is the symbol written into cell j of tape i during the simulation, or $T_i(x, k-1, j)$ if this cell has not been passed. For the simulation we need to maintain the current state and the positions of the heads of all tapes. By Lemma 2.3 these positions can be numbers between 1 and $2^{O(r(n)\cdot s(n))}$ (recall that $r(n)\cdot s(n) \in \Omega(\log n)$). Hence, $O(r(n)\cdot s(n))$ space suffices to store the current state and head positions on all tapes. We additionally need space to compute $S(x, k-1)$, $P_i(x, k-1)$ and $T_i(x, k-1, j)$. Since S, P_i and T_i can be computed in constant space for $k = 0$, it follows by induction on k that $O(k\cdot r(n)\cdot s(n))$ space suffices to compute S, P_i and T_i for $k \geq 1$.

Finally, to determine the final state of M on input x, we iteratively compute $S(x, 1), S(x, 2), \ldots$ until $S(x, k)$ is a final state for some k. Since on an input x of length n, M performs at most $r(n)$ head reversals, we have $k \leq r(n)$. Hence, the final state of M on input x can be determined in space $O(r(n)^2 \cdot s(n))$. □

We write Polylogspace to denote the complexity class that consists of all decision problems that can be solved in space $(\log n)^{O(1)}$. Using Lemma 4.8, Theorems 4.6 and 3.1, one easily obtains the following:

Theorem 4.9. (a) $\mathrm{ST}(O(1), O(\log n), O(1)) \subseteq \mathrm{Logspace} \subseteq \mathrm{ST}(O(\log n), O(1), O(1)) \subseteq \mathrm{DSpace}((\log n)^2)$
(b) $\mathrm{Polylogspace} = \mathrm{ST}((\log n)^{O(1)}, O(1), O(1)) = \mathrm{ST}((\log n)^{O(1)}, (\log n)^{O(1)}, O(1))$.

Proof. (a): The first inclusion is due to Lemma 4.8; the second and the third inclusion are due to Theorem 4.6.
(b): This immediately follows from Theorems 4.6 and 3.1. □

Later on, in Remark 4.12 we will see that the inclusion of $\mathrm{ST}(O(1), O(\log n), O(1))$ in Logspace is, in fact, *strict*.

Remark 4.10. Let us note that our approach for proving Lemma 4.8 does not work for nondeterministic or randomized machines. In fact, a nondeterministic analogue of Lemma 4.8 of the form

$$\mathrm{NST}(r, s, O(1)) \subseteq \mathrm{NSpace}(r^{O(1)} \cdot s^{O(1)}) \qquad (\star\star)$$

seems rather unlikely, since this would imply that NP $\subseteq$ Polylogspace. To see this, recall from Corollary 4.4 that NP $=$ $\mathrm{NST}(O(1), O(\log n), O(1))$. Thus, $(\star\star)$ would imply that NP is included in $\mathrm{NSpace}((\log n)^{O(1)})$ which, due to Savitch's theorem [20] is equal to $\mathrm{DSpace}((\log n)^{O(1)}) =$ Polylogspace.

4.3. Circuit complexity classes

Recall that NC^i is the class of all decision problems that can be solved by uniform families of circuits of size $n^{O(1)}$ and depth $(\log n)^i$, that consist of bounded fan-in $\{\wedge, \vee, \neg\}$-gates. AC^i denotes the analogous class where gates of unbounded fan-in are allowed. Thus, for all $i \geq 0$, we have $\mathrm{NC}^i \subseteq \mathrm{AC}^i \subseteq \mathrm{NC}^{i+1}$. Furthermore, NC denotes the union of the classes NC^i, for all $i \geq 0$. The next lemma shows that $(\log n)^i$ head reversals and constant internal memory suffice to simulate all NC^i circuits, and that this is optimal in the following sense: just $o(\log n)$ head reversals are too weak for simulating AC^0 circuits, even if internal memory may get as large as $O(\sqrt[4]{N}/\log N)$.

Lemma 4.11. (a) $\mathrm{NC}^i \subseteq \mathrm{ST}(O((\log n)^i), O(1), O(1))$, *for all* $i \geq 0$.
(b) Short-Set-Equality $\in \mathrm{AC}^0$.
(c) $\mathrm{AC}^0 \not\subseteq \mathrm{ST}(o(\log n), O(\sqrt[4]{N}/\log n), O(1))$.

Proof. (a): It is known that $\mathrm{NC}^i \subseteq \mathrm{DSpace}((\log n)^i)$ (cf., e.g., [1, Proposition 4.2]). Due to Theorem 4.6 we have $\mathrm{DSpace}((\log n)^i) \subseteq \mathrm{ST}(O((\log n)^i), O(1), O(1))$, and thus the claim follows.

(b): It is known (cf., [11]) that uniform AC^0 can be characterized as the class of all string-languages that are definable by a sentence of *first-order logic* with built-in *Bit*-predicate, FO[$<$, *Bit*], for short (for an introduction to first-order logic and its relation to complexity theory see, e.g., the textbook [12]). To prove (b), it therefore suffices to find an FO[$<$, *Bit*]-sentence ψ that is satisfied exactly by those strings that are "yes"-instances of the Short-Set-Equality problem. To construct such a sentence, we use the result (cf. e.g. [6]; or [5] for a purely logical proof) that FO[$<$, *Bit*] has the "polylog counting capability", i.e., for every fixed $i \in \mathbb{N}$ there is an FO[$<$, *Bit*]-formula $\varphi^i_{count}(x, Y)$ such that for all $n \in \mathbb{N}$, $x \in \{0, \ldots, n\}$, and $Y \subseteq \{0, \ldots, n\}$, the structure $(\{0, \ldots, n\}, <, \mathit{Bit}, x, Y)$ satisfies the formula $\varphi^i_{count}(x, Y)$ if, and only if, $x = |Y| \leq (\log n)^i$.

Recall that an input instance for the Short-Set-Equality problem is a string of the form

$$v_1 \# \cdots \# v_m \# v'_1 \# \cdots \# v'_m \#$$

where $m \geq 1$ and each v_i and v'_j is a $\{0, 1\}$-string of length $2 \cdot \log m$. Using the counting formula $\varphi^1_{count}(x, Y)$, it is straightforward to construct an FO[$<$, *Bit*]-sentence $\psi_{instance}$ that is satisfied by exactly those strings that are valid input instances of Short-Set-Equality. In addition, we can construct an FO[$<$, *Bit*]-sentence ψ_{yes} that is satisfied by a valid input instance if, and only if, this is a "yes"-instance. To this end, the formula ψ_{yes} just has to express that for every position x carrying a #-symbol in the first half, there is an according position y in the second half (and vice versa), such that the $\{0, 1\}$-strings of length $2 \cdot \log m$ to the left of x and y, respectively, are equal — and equality of substrings of length $2 \cdot \log m$ can easily be checked when using the counting formula φ^1_{count}.

Finally, the desired FO[$<$, *Bit*]-sentence ψ that defines Short-Set-Equality is chosen as the conjunction of the two sentences $\psi_{instance}$ and ψ_{yes}. This completes the proof of (b).

(c): Follows immediately from (b) and Theorem 2.7. □

Remark 4.12. In particular, as an application of Lemma 4.11 and Theorem 2.7 we obtain that $\mathrm{ST}(O(1), O(\log n), O(1)) \subsetneq$ Logspace, because Short-Set-Equality belongs to $\mathrm{AC}^0 \subseteq$ Logspace, but not to $\mathrm{ST}(O(1), O(\log n), O(1))$.

In [19], Pippenger showed that NC is precisely the class of languages recognized by deterministic multi-tape Turing machines that *simultaneously* have time-bound $n^{O(1)}$ and perform at most $(\log n)^{O(1)}$ head reversals on all their tapes. The inclusion "$\subseteq$" follows by a rather direct construction; the proof of the opposite inclusion "$\supseteq$" is more intricate and is obtained by showing that a Turing machine which simultaneously has time-bound $T(n)$ and performs only $r(n)$ head reversals, can be simulated by a uniform family of circuits of depth $O(r(n) \cdot (\log T(n))^4)$ and size $O(r(n) \cdot T(n)^{O(1)})$. This simulation result was improved by Parberry [17] to:

Theorem 4.13 (*Parberry [17]*). *A deterministic k-tape Turing machine which simultaneously has space-bound $S(n)$ and performs at most $r(n)$ head reversals on all its tapes, can be simulated by a uniform family of circuits (with $\{\wedge, \vee, \neg\}$-gates of bounded fan-in) of depth $O(r(n)\cdot(\log S(n))^2)$ and width $O(S(n)^k)$.*

Note that the present paper's (r, s, t)-bounded Turing machines, which are used to define the complexity classes $\mathrm{ST}(r, s, t)$, are different from Pippenger's and Parberry's machines, as the latter are *simultaneously* bounded in head reversals and either time or space on *all* tapes, whereas our (r, s, t)-bounded machines have some tapes (namely, the t external memory tapes) which are unrestricted in size but restricted in head reversals, and some further tapes (namely, the internal memory tapes) which are restricted in size but unrestricted in terms of head reversals. However, by using Theorem 3.1 and Lemma 2.3, Parberry's Theorem 4.13 immediately implies that $(r, s, O(1))$-bounded deterministic Turing machines can be simulated by circuits of the following kind:

Corollary 4.14. *Let r and s be functions such that $r(n)\cdot s(n) \in \Omega(\log n)$. Then, every language in $\mathrm{ST}(r, s, O(1))$ can be decided by a uniform family of circuits (with $\{\wedge, \vee, \neg\}$-gates of bounded fan-in) of depth $O(r(n)^3\cdot s(n)^3)$ and size $2^{O(r(n)\cdot s(n))}$.*

Proof. By Theorem 3.1, $\mathrm{ST}(r, s, O(1)) \subseteq \mathrm{ST}(O(r\cdot s), O(1), O(1))$.

Furthermore, we know from Lemma 2.3 that an $(O(r\cdot s), O(1), O(1))$-bounded Turing machine uses space $\leq 2^{O(r(n)\cdot s(n))}$ on *all* its tapes and therefore simultaneously has space-bound $S(n) \in 2^{O(r(n)\cdot s(n))}$ and performs at most $O(r(n)\cdot s(n))$ head reversals on all its tapes. Thus, Theorem 4.13 gives us a uniform family of circuits of depth $O(r(n)^3\cdot s(n)^3)$ and width $S(n)^{O(1)} \subseteq 2^{O(r(n)\cdot s(n))}$. Since the *size* of a circuit is bounded by the product of its *depth* and its *width*, the claim follows and the proof of Corollary 4.14 is complete. □

5. Hierarchies of external memory classes

In this section we prove hierarchies within the $(\mathrm{R})\mathrm{ST}(\cdots)$ classes. First, in Section 5.1, we consider classes in which arbitrarily many external memory tapes are available; afterwards, in Section 5.2, we turn to classes with only one external memory tape.

5.1. Hierarchies of classes with arbitrarily many tapes

Chen and Yap have observed in [4] that their Theorem 4.6, together with the *space hierarchy theorem* (cf., e.g., the textbook [2]), leads to a strict hierarchy of reversal complexity classes. Using Lemma 4.8, this leads to:

Proposition 5.1. *Let r, s, and R be functions such that $r(n)\cdot s(n) \in \Omega(\log n)$, $R(n) \in \omega(r(n)^2\cdot s(n))$, and R reversal-constructible and space-constructible. Then,*

$$\mathrm{ST}(O(r), O(s), O(1)) \subsetneq \mathrm{ST}(O(R), O(1), O(1)).$$

Proof.

$$\begin{aligned} \mathrm{ST}(O(r), O(s), O(1)) &\subseteq \mathrm{DSPACE}(r^2\cdot s) && \text{(Lemma 4.8)}\\ &\subsetneq \mathrm{DSPACE}(R) && \text{(space hierarchy theorem)}\\ &\subseteq \mathrm{ST}(O(R), O(1), O(1)) && \text{(Theorem 4.6).} \quad \square \end{aligned}$$

Remark 5.2. For example, the above proposition in particular tells us for all $k \geq 1$ that

$$\mathrm{ST}(O((\log n)^k), O(\log n), O(1)) \subsetneq \mathrm{ST}(O((\log n)^{2k+2}), O(1), O(1)).$$

Note that Proposition 5.1 does not apply to classes where only $o(\log n)$ head reversals are available. To also treat such cases, we consider padded versions of the Multiset-Equality problem to transfer the lower bound of Theorem 2.7 into the following separations for classes with $o(\log n)$ head reversals:

Lemma 5.3. *Let r and s be functions such that $r(n) \in o(\log n) \cap \omega(1)$ and*

(i) *given an input string of length n, a string of length $r(n)$ can be produced by a deterministic $(O(r(n)), O(s(n)), O(1))$-bounded Turing machine, and*
(ii) *given an input of length $\tilde{n}$, a string of length n, where n is the smallest number with $r(n) \leq \log \tilde{n} \leq r(n{+}1)$, can be produced by a deterministic $(o(\log \tilde{n}), O(\sqrt[5]{\tilde{n}}), O(1))$-bounded Turing machine.*

Then, there is a decision problem that belongs to $\mathrm{ST}(O(r(n{+}1)), O(s(n{+}1)), O(1))$, but not to $\mathrm{RST}(o(r(n)), O(\sqrt[5]{2^{r(n)}}), O(1))$.

Proof. Consider the decision problem

r-Padded-Multiset-Equality
Instance: $v_1\#\cdots v_m\#v'_1\#\cdots v'_m\#w$, where $m \geq 1, v_1, \ldots, v_m, v'_1, \ldots, v'_m \in \{0, 1\}^*, w \in \{1\}^*$ such that for the total input length n and for $\tilde{n} := n-|w|$ we have $r(n) \leq \log \tilde{n} \leq r(n{+}1)$.
Problem: Decide if the multisets $\{v_1, \ldots, v_m\}$ and $\{v'_1, \ldots, v'_m\}$ are equal.

To show that r-Padded-Multiset-Equality does

not belong to $\mathrm{RST}(o(r(n)), O(\sqrt[5]{2^{r(n)}}), O(1))$,

we give a reduction from Multiset-Equality to r-Padded-Multiset-Equality: Given an input of length $\tilde{n}$ of the form $v_1\#\cdots v_m\#v'_1\#\cdots v'_m\#$ for the Multiset-Equality problem, we proceed as follows: First, we search for a number n such that $r(n) \leq \log\tilde{n} \leq r(n{+}1)$. Due to assumption (ii), this can be achieved with $o(\log\tilde{n})$ head reversals on external memory tapes and $O(\sqrt[5]{\tilde{n}})$ internal memory space. Afterwards, we pad the input to a string of length n to obtain an input instance for r-Padded-Multiset-Equality. Thus, if r-Padded-Multiset-Equality belonged to $\mathrm{RST}(o(r(n)), O(\sqrt[5]{2^{r(n)}}), O(1))$, then Multiset-Equality could be solved by a randomized $(o(\log\tilde{n}), O(\sqrt[5]{\tilde{n}}), O(1))$-bounded Turing machine (to see this, recall that $r(n) \leq \log\tilde{n}$ and $\sqrt[5]{2^{r(n)}} \leq \sqrt[5]{2^{\log\tilde{n}}} = \sqrt[5]{\tilde{n}}$). This, however, is a contradiction to Theorem 2.7 (a).

To see that r-Padded-Multiset-Equality belongs to $\mathrm{ST}(O(r(n{+}1)), O(s(n{+}1)), O(1))$, we construct a Turing machine M which, in a first phase, tests if the input is an admissible instance for r-Padded-Multiset-Equality as follows: M checks if the input is of the form $v_1\#\cdots v_m\#v'_1\#\cdots v'_m\#w$, for some $m \geq 1$, such that $w \in \{1\}^*$ and $v_i, v'_j \in \{0,1\}^*$, for all $i, j \leq m$ (this can be done with a constant number of head reversals on two external memory tapes). Then, M checks for the total input length n and the number $\tilde{n} := n - |w|$, that $r(n) \leq \log\tilde{n} \leq r(n{+}1)$ (due to assumption (i), this can be done with $O(r(n{+}1))$ head reversals on external memory tapes and internal memory space $O(s(n{+}1))$). This completes M's first phase. Provided that the input is an admissible instance, M then sorts each of the two lists $v_1\#\cdots v_m\#$ and $v'_1\#\cdots v'_m\#$ of $\{0,1\}$-strings in ascending lexicographic order. Due to [4, Lemma 7], this can be achieved with constant internal memory space and $O(\log\tilde{n}) \leq O(r(n{+}1))$ head reversals on two external memory tapes. Afterwards, a single scan of the two sorted lists in parallel suffices to decide if the multisets $\{v_1, \ldots, v_m\}$ and $\{v'_1, \ldots, v'_m\}$ are equal. □

As a consequence of Lemma 5.3 we obtain, for example, the following separation result:

Theorem 5.4. *For every $k \geq 2$,*

$$\mathrm{ST}\left(O\left(\sqrt[k]{\log n}\right), O(\log n), O(1)\right) \not\subseteq \mathrm{RST}\left(o\left(\sqrt[k]{\log n}\right), (\log n)^{O(1)}, O(1)\right).$$

In particular, this implies the following hierarchy of classes for all $k \geq 2$:

$$(\mathrm{R})\mathrm{ST}\left(O\left(\sqrt[k+1]{\log n}\right), O(\log n), O(1)\right) \subsetneq (\mathrm{R})\mathrm{ST}\left(O\left(\sqrt[k]{\log n}\right), O(\log n), O(1)\right).$$

Proof. The second statement obviously follows from the first statement. To prove the first statement, use Lemma 5.3 for the functions $r(n) := \sqrt[k]{\log n}$ and $s(n) := \log n$. Obviously, $r(n) \in o(\log n) \cap \omega(1)$. Furthermore, assumption (i) of Lemma 5.3 is satisfied, because on an input of length n, internal memory of size $s(n) = \log n$ is available and thus the binary representations of n, $\log n$, and $\sqrt[k]{\log n}$ can be computed in internal memory. Then, a string of length $r(n) = \sqrt[k]{\log n}$ can be produced with a single scan on an external memory tape. Concerning assumption (ii) of Lemma 5.3, we need to find a Turing machine which, on an input of length $\tilde{n}$, constructs a string of length n, where $\sqrt[k]{\log n} = r(n) \leq \log\tilde{n} \leq \sqrt[k]{\log(n{+}1)}$, i.e., $n \leq 2^{((\log\tilde{n})^k)} \leq n{+}1$. To achieve this, we may use internal memory of size up to $\sqrt[5]{\tilde{n}}$. Note that with internal memory of that size, the binary representations of numbers of size up to $2^{\sqrt[5]{\tilde{n}}} \geq 2^{((\log\tilde{n})^k)}$ can be handled in internal memory. Thus, the binary representation of n can be computed in internal memory, and afterwards a string of length n can be produced with a single scan on an external memory tape.

Therefore, the assumptions of Lemma 5.3 are satisfied, and Lemma 5.3 tells us that $\mathrm{ST}(O(\sqrt[k]{\log n}), O(\log n), O(1)) \not\subseteq \mathrm{RST}(o(\sqrt[k]{\log n}), \sqrt[5]{2^{r(n)}}, O(1))$. Theorem 5.4 follows, since $\sqrt[k+1]{\log n} \in o(\sqrt[k]{\log n})$ and $\sqrt[5]{2^{r(n)}} \in 2^{\omega(\log\log n)}$, and thus is larger than any internal memory bound in $(\log n)^{O(1)}$. □

A visualization of the hierarchy results from Section 5.1 is shown in Fig. 2.

5.2. Hierarchies of classes with a single external memory tape

When considering $\mathrm{ST}(\cdots)$ classes where only *one* external memory tape is available, strong results and methods from *communication complexity* can be used to prove lower bounds. This technique has already been used in [9] to obtain bounds on the data complexity of query evaluation problems on data streams and to obtain a hierarchy of $\mathrm{ST}(r, s, 1)$ classes for the special case where r is *constant*. This subsection's main result generalizes the hierarchy of [9] to non-constant functions r. The method of choice, again, is to use appropriate lower bound results for communication complexity. This subsection's main result is

Theorem 5.5. *For every logspace-computable function r and for all classes S of functions $s\colon \mathbb{N} \to \mathbb{N}$ we have:*

(a) *If $r(n) \in o(n/(\log n)^2)$ and $O(\log n) \subseteq S \subseteq o\left(\frac{n}{r(n)\log n}\right)$, then $\mathrm{ST}(r(n), S, 1) \subsetneq \mathrm{ST}(r(n){+}1, S, 1)$.*

(b) *If $r(n)^3 \in o(n/(\log n)^2)$ and $O(\log n) \subseteq S \subseteq o\left(\frac{n}{r(n)^3\log n}\right)$, then $\mathrm{RST}(r(n), S, 1) \subsetneq \mathrm{RST}(r(n){+}1, S, 1)$.*

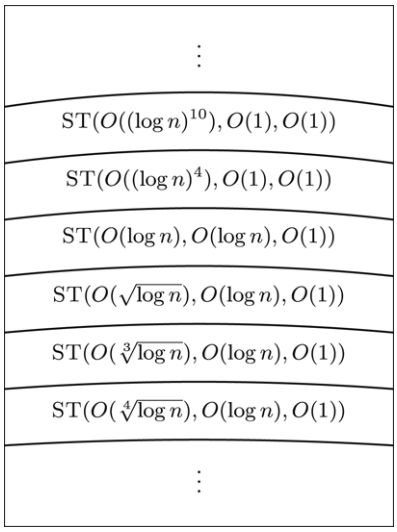


Fig. 2. Visualization of the hierarchy results from Section 5.1.

Before proving Theorem 5.5, we introduce some basic definitions and a strong result from communication complexity. Let X, Y be finite sets and $f : X \times Y \to \{0, 1\}$ a Boolean function. In Yao's [24] basic model of communication, Alice gets an input $x \in X$, Bob gets an input $y \in Y$, and together they wish to evaluate $f(x, y)$ by exchanging messages according to a fixed *protocol* $\mathcal{P}$. The *cost* of $\mathcal{P}$ is the worst case number of bits that have to be communicated, so that Alice and Bob both know $f(x, y)$. *Randomization* can be introduced by giving each player access to his or her own random string. Then, we say that $\mathcal{P}$ *computes* f *with* $1/3$*-error* if for all inputs $(x, y) \in X \times Y$, $\mathcal{P}$ computes the correct value $f(x, y)$ with probability $\geq 2/3$.

A protocol has k *rounds*, for some $k \in \mathbb{N}$, if Alice and Bob alternately exchange at most k messages. The *deterministic* k*-round communication complexity of* f is defined as the minimum cost of a deterministic k-round protocol that computes f. The *randomized* k*-round communication complexity of* f is the minimum cost of a randomized k-round protocol that computes f with $1/3$-error.

To prove Theorem 5.5, we consider the *pointer jumping function* $\mathrm{pj}_{\hat{n},k}$ which, for given parameters $\hat{n}, k \in \mathbb{N}$, is defined as follows: $\mathrm{pj}_{\hat{n},k}(x, y) := 1$ if $x = w_0 \cdots w_{\hat{n}-1}$, $y = w_{\hat{n}} \cdots w_{2\hat{n}-1}$, where $w_i \in \{0, 1\}^{\log(2\cdot\hat{n})}$, and there exist indices $j_1, \ldots, j_k$ such that[3] $w_1 = \mathrm{bin}(j_1)$, $w_{j_i} = \mathrm{bin}(j_{i+1})$ and w_{j_k} has an even number of 1s; and $\mathrm{pj}_{\hat{n},k}(x, y) := 0$ otherwise.

We will use the following lower bounds on the communication complexity of $\mathrm{pj}_{\hat{n},k}$:

Theorem 5.6 (*Nisan and Wigderson [15]*). *Let* $k, \hat{n} \in \mathbb{N}$.

If $k < \hat{n}/\log\hat{n}$*, then the deterministic* k*-round communication complexity of* $\mathrm{pj}_{\hat{n},k+1}$ *is* $\Omega(\hat{n})$*. Furthermore, if* $k < \sqrt[3]{\hat{n}/\log\hat{n}}$*, then the randomized* k*-round communication complexity of* $\mathrm{pj}_{\hat{n},k+1}$ *is* $\Omega(\hat{n}/k^2)$.

Theorem 5.5 now is obtained as an immediate consequence of the following lemma, which gives upper and lower bounds for the language PJ_{r+1}, defined for a given function $r : \mathbb{N} \to \mathbb{N}$ via

$$\mathrm{PJ}_{r+1} := \{\, 1^m\#xy : m \geq 1,\ x, y \in \{0, 1\}^{m\cdot\hat{n}} \text{ with } \hat{n} := 2^{m-1}, \text{ and } \mathrm{pj}_{\hat{n},r(2m\hat{n}+m+1)+1}(x, y) = 1 \,\}.$$

Lemma 5.7. (a) *For all logspace-computable functions* $r : \mathbb{N} \to \mathbb{N}$*, we have*
$\mathrm{PJ}_{r+1} \in \mathrm{ST}(r(n)+1, O((\log r(n)) + \log n), 1)$.
(b) *Let* $r, s : \mathbb{N} \to \mathbb{N}$ *such that* $r(n) \in o(n/(\log n)^2)$ *and* $r(n)\cdot s(n) \in o(n/\log n)$.
Then, $\mathrm{PJ}_{r+1} \notin \mathrm{ST}(r(n), s(n), 1)$.
(c) *Let* $r, s : \mathbb{N} \to \mathbb{N}$ *such that* $r(n)^3 \in o(n/(\log n)^2)$ *and* $r(n)^3\cdot s(n) \in o(n/\log n)$.
Then, $\mathrm{PJ}_{r+1} \notin \mathrm{RST}(r(n), s(n), 1)$.

Proof. (a): Let $w = 1^m\#w_0w_1 \ldots w_{2^m-1}$ be an input of length n, where $w_i \in \{0, 1\}^m$ for all $i \in \{0, 1, \ldots, 2^m-1\}$. To decide whether $w \in \mathrm{PJ}_{r+1}$, we just have to determine $j_1, j_2, \ldots, j_k$, $k = r(n) + 1$, one after the other, where $\mathrm{bin}(j_i) = w_{j_{i-1}}$ is determined from j_{i-1} with at most one head reversal on the external memory tape. In internal memory we maintain the value of m ($\log n$ bits), the current value of j_i ($\log n$ bits), and a counter ($\log r(n)$ bits) which, at the beginning, is initialized to $r(n)$ and decreased in every step. Note that we do not need any head reversal on the external memory tape to determine w_{j_1} from w_0. Therefore we need at most $r(n)$ head reversals and $O((\log r(n)) + \log n)$ internal memory space.

[3] Here, we use $\mathrm{bin}(j)$ to denote the binary representation of j of length $\log(2\cdot\hat{n})$.

(b): For contradiction, we assume that there is a deterministic $(r(n), s(n), 1)$-bounded Turing machine M that decides PJ_{r+1}. Let Q be the set of states of M. For $m \in \mathbb{N}$ let $\hat{n} := 2^{m-1}$, $n := 2m\hat{n} + m + 1$ and $k := r(n)$. Then, M leads to a deterministic k-round communication protocol for $\mathrm{pj}_{\hat{n},k+1}$ as follows: Let $x \in \{0,1\}^{m\cdot\hat{n}}$ be the input for Alice and $y \in \{0,1\}^{m\cdot\hat{n}}$ be the input for Bob. Alice simulates M on input $1^m\#xy$ until M reads cell $p := |1^m\#x| + 1$ on its external memory tape. Then, Alice sends the current state of M ($\log|Q|$ bits) and the contents of M's internal memory tapes ($O(s(n))$ bits) to Bob. Bob continues the simulation until M accesses cell $p-1$, sends the current state of M and the contents of its internal memory tapes to Alice, and so on. Since M is $(r(n), s(n), 1)$-bounded, the head of the external memory tape passes cell p, or $p-1$ respectively, at most $r(n) = k$ times. So the number of rounds of the protocol is k and the total communication is at most $r(n)(O(s(n)) + \log|Q|)$, which is of size $o(n/\log n) = o(\hat{n})$ by the assumption of (b). Moreover, $r(n) \in o(n/(\log n)^2) = o(\hat{n}/\log\hat{n})$. Hence, for sufficiently large m we obtain a k-round-protocol for $\mathrm{pj}_{\hat{n},k+1}$ with communication $o(\hat{n})$, where $k < \hat{n}/\log\hat{n}$. But this contradicts Theorem 5.6's statement on deterministic k-round communication complexity.

(c): This can be shown in the same way as (b), when using Theorem 5.6's statement on randomized k-round communication complexity. □

Note that Theorem 5.5 is an immediate consequence of Lemma 5.7.

6. Conclusion

The present paper's main results are (a) a trade-off between internal memory space and external memory head reversals, stating that internal memory can be compressed from size $s(n)$ to $O(1)$ at the expense of adding an extra factor $s(n)$ to the external memory head reversals (see Section 3), (b) correspondences between the (R,N)ST$(\cdots)$ classes and "classical" time-bounded, space-bounded, reversal-bounded, and circuit complexity classes (see Section 4), and (c) hierarchies of (R)ST$(\cdots)$-classes in terms of increasing numbers of head reversals on external memory tapes (see Section 5). Visualisations of some of our results can be found in Figs. 1 and 2.

An intriguing future task is to develop techniques for proving lower bounds for appropriate decision problems in a setting where $\Omega(\log n)$ head reversals and several external memory tapes are available. Of course, particular separating problems are immediately obtained from the space hierarchy theorem via Proposition 5.1 (see Remark 5.2). However, finding lower bounds for "*natural*" decision problems can be expected to be rather difficult, since we know from Theorem 4.9 that $\textsc{Logspace} \subseteq \mathrm{ST}(O(\log n), O(1), O(1))$.

To conclude the paper, let us comment on the open questions listed in the conclusion of Chen and Yap's article [4]. One question was whether one can "speedup" reversal complexity by a constant factor, i.e., whether, for $c > 1$, $c \cdot r(n)$ reversals can be simulated by just $r(n)$ head reversals. Theorem 5.5 shows that there is no such speedup for ST and RST-classes with just *one* external memory tape, as long as r is in $o(n/(\log n)^2)$, respectively, in $o(\sqrt[3]{n/(\log n)^2})$.

Another question of [4] was whether $r(n)$ reversals on two external memory tapes can be simulated by $r(n)^{O(1)}$ reversals on a single external memory tape. In general, this is not the case. For example, Theorem 4.6 implies that $\textsc{Logspace}$ is included in the class $\mathrm{ST}(O(\log n), O(1), 2)$. However, it is not included in $\mathrm{ST}(O((\log n)^{O(1)}), O(1), 1)$. To see this, let $S(x)$ be the subset of $\{1, \ldots, n\}$ represented by the characteristic string x (i.e., $x = x_1 \cdots x_n$ is a bit-string of length n, where $x_i = 1$ if and only if $i \in S(x)$). Let

$$L_{Disj} := \{x\#y \mid x, y \in \{0,1\}^n \text{ and } S(x) \cap S(y) = \emptyset\}.$$

Then, L_{Disj} is easily seen to be in $\textsc{Logspace}$. On the other hand, [9, Proposition 4.3] says that $L_{Disj} \notin \mathrm{ST}(r(n), s(n), 1)$, whenever $r(n)\cdot s(n) \in o(n)$. In particular, $L_{Disj} \notin \mathrm{ST}(O((\log n)^{O(1)}), O(1), 1)$.

References

[1] J.L. Balcázar, J. Díaz, J. Gabarró, Structural Complexity II, Springer-Verlag, 1990.
[2] J.L. Balcázar, J. Díaz, J. Gabarró, Structural Complexity I, 2nd ed., Springer-Verlag, 1995.
[3] P. Beame, T.S. Jayram, A. Rudra, Lower bounds for randomized read/write stream algorithms, in: Proc. STOC'07, ACM, 2007, pp. 689–698.
[4] J. Chen, C.-K. Yap, Reversal complexity, SIAM Journal on Computing 20 (4) (1991) 622–638.
[5] A. Durand, C. Lautemann, M. More, Counting results in weak formalisms, Technical Report 1998-14, Université de Caen, France, 1998.
[6] R. Fagin, M. Klawe, N. Pippenger, L. Stockmeyer, Bounded depth, polynomial size circuits for symmetric functions, Theoretical Computer Science 36 (1985) 239–250.
[7] M. Grohe, A. Hernich, N. Schweikardt, Randomized computations on large data sets: Tight lower bounds, in: Proc. PODS'06, ACM Press, 2006, pp. 243–252.
[8] M. Grohe, C. Koch, N. Schweikardt, The complexity of querying external memory and streaming data, in: Proc. FCT'05, in: Lecture Notes in Computer Science, vol. 3623, 2005, pp. 1–16.
[9] M. Grohe, C. Koch, N. Schweikardt, Tight lower bounds for query processing on streaming and external memory data, Theoretical Computer Science 380 (1–2) (2007) 199–217.
[10] M. Grohe, N. Schweikardt, Lower bounds for sorting with few random accesses to external memory, in: Proc. PODS'05, ACM Press, 2005, pp. 238–249.
[11] N. Immerman, Expressibility and parallel complexity, SIAM Journal on Computing 18 (1989) 625–638.
[12] N. Immerman, Descriptive Complexity, Springer-Verlag, 1998.
[13] M. Liśkiewicz, On the relationship between deterministic time and deterministic reversal, Information Processing Letters 45 (3) (1993) 143–146.
[14] U. Meyer, P. Sanders, J.F. Sibeyn (Eds.), Algorithms for Memory Hierarchies, in: Lecture Notes in Computer Science, vol. 2625, Springer-Verlag, 2003.
[15] N. Nisan, A. Wigderson, Rounds in communication complexity revisited, SIAM Journal on Computing 22 (1) (1993) 211–219.
[16] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.

[17] I. Parberry, An improved simulation of space and reversal bounded deterministic turing machines by width and depth bounded uniform circuits, Information Processing Letters 24 (6) (1987) 363–367.
[18] I. Parberry, A note on nondeterminism in small, fast parallel computers, IEEE Transactions on Computers 38 (5) (1989) 766–767.
[19] N. Pippenger, On simultaneous resource bounds, in: Proc. FOCS'79, 1979, pp. 307–311.
[20] W.J. Savitch, Relationships between nondeterministic and deterministic tape complexities, Journal of Computer and Systems Sciences 4 (1970) 177–192.
[21] N. Schweikardt, Machine models and lower bounds for query processing, in: Proc. PODS'07, ACM Press, 2005, pp. 41–52.
[22] J.F. Vitter, External memory algorithms and data structures: Dealing with massive data, ACM Computing Surveys 33 (2001) 209–271.
[23] K. Wagner, G. Wechsung, Computational Complexity, VEB Deutscher Verlag der Wissenschaften, 1986.
[24] A.C.C. Yao, Some complexity questions related to distributive computing, in: Proc. STOC'79, ACM Press, 1979, pp. 209–213.